IMPORTANT: Always take that case into consideration where we will have only 1 element in the array.

BINARY SEARCH 1

→ Any Searching problem has 2 components, namely target and search space.

→ If I have to search for a word in newspaper then newspaper is search space and word will be my target.

→

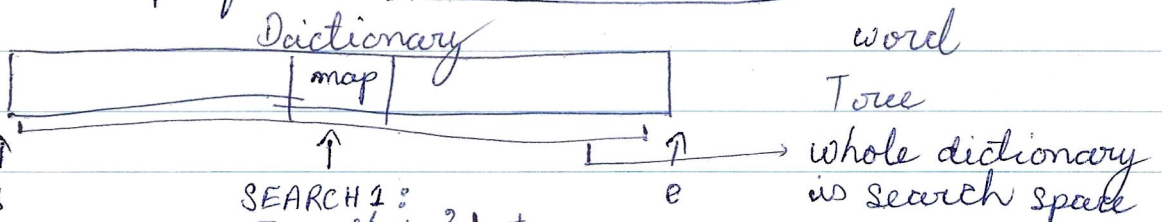
	Search Space		Target
Word in newspaper	Word	⟺	Newspaper
Word in dictionary	Dictionary		Word
Name in Contact List	Contact List		Name

→ Is there any difference between finding a word in newspaper and dictionary?

Newspaper – Words are unsorted
Dictionary – Words are sorted.

→ Newspaper – To search a word in newspaper, one will do linear search. O(N)

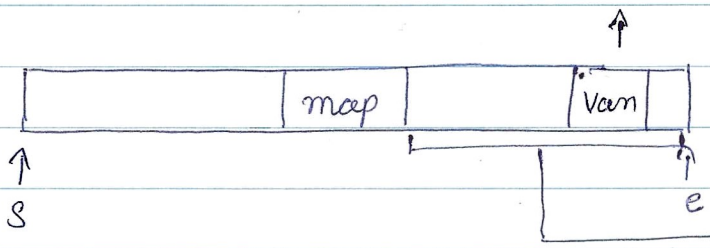
→ Dictionary – To search a word in dictionary, one performs random search.



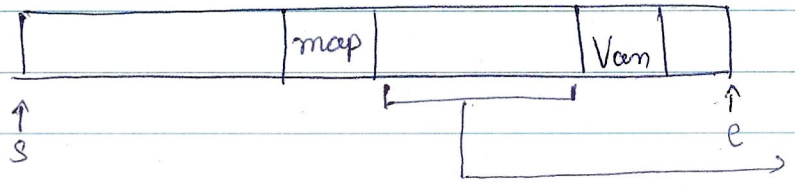
RANDOM SEARCH

BINARY SEARCH 1

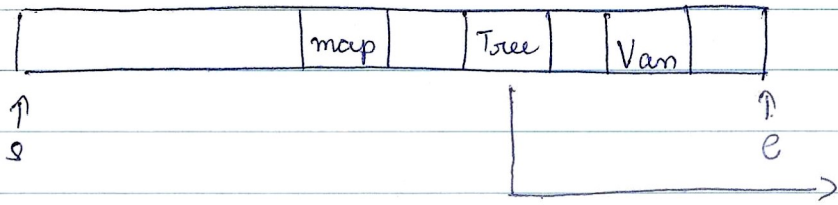
RANDOM SEARCH



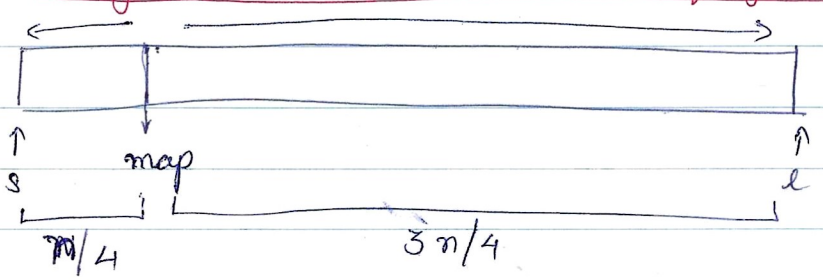
RANDOM SEARCH



RANDOM SEARCH



Does random search saves time? Yes
Is it a best way to search? No
Why No? Why RANDOM SEARCH is not good?



Suppose, we searched for word tree but firstly we found 'map'. Now if 'tree' lies on left of map then we will search n/4 search space or 1/4th of search space and if it lies on right then we need to search 3n/4 search space

BINARY SEARCH 1

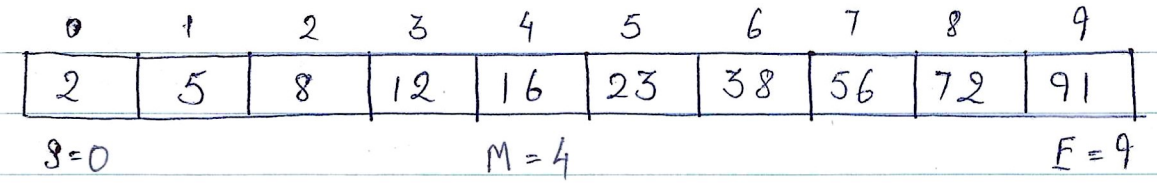
or 3/4 th of the search space. By randomly picking a word for search, search space is not getting evenly distributed which is a drawback of random search. This is called unbalanced split.

NOTE : Because of the problem of unbalanced split, binary search is preferred which provides us balanced split or splits of equal size inform of search space.

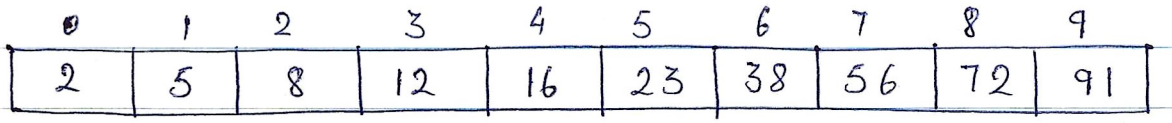
Example of binary Search 10:15

Example 1

Search 23

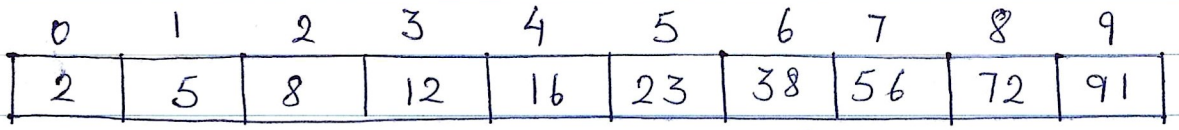


16 < 23



56 > 23

S = M + 1



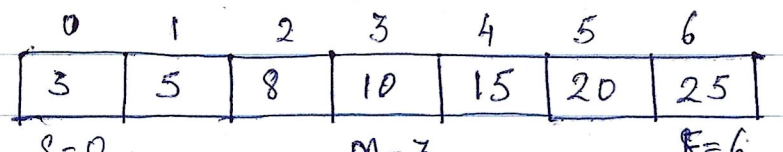
23 = 23

E = M - 1

Return 5 return M

Example 2

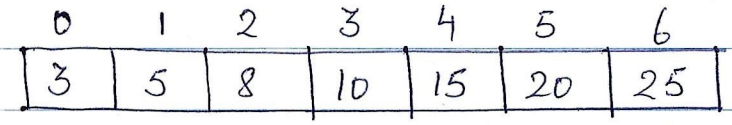
Search 15



10 < 15

BINARY SEARCH 1

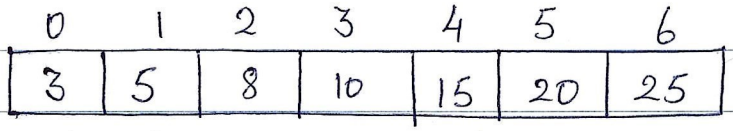
Search 15



S=4 M=5 E=6

20 > 15

S = M + 1



S=4

E=4

M=4

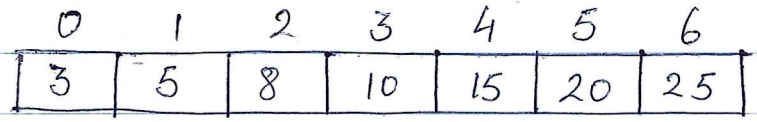
15 = 15

E = M - 1

Return 4 return M

Example 3

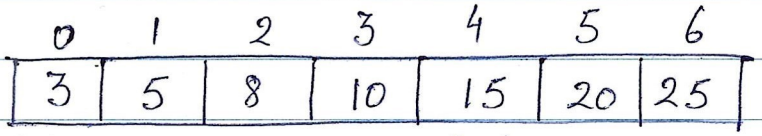
Search 14



S=0 M=3 E=6

10 < 14

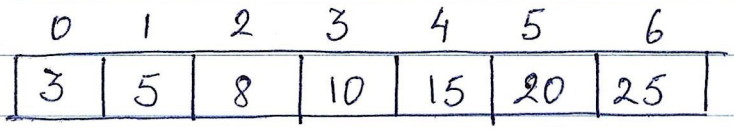
S = M + 1



S=4 M=5 E=6

20 > 14

E = M - 1



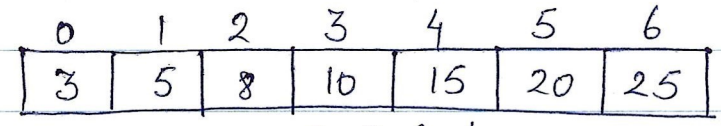
S=4

E=4

M=4

15 > 14

E = M - 1



E=3 S=4

S > E, terminate the loop and return -1. Element doesn't exist

Return -1

BINARY SEARCH 1

How mid is determined in binary search?

Simplest way:

$$M = \frac{(S + E)}{2}$$

But when S and E are very high values, we could face the issue of overflow and could lead to -ve values for indices of M. In this case we could use the formula below.

$$M = S + ((E - S)/2)$$

Search 15

0	1	2	3	4	5	6
3	5	8	10	15	20	25

S=0 M=3 E=6

M = (0+6)/2 = 3

M = 0 + ((6-0)/2)
= 0 + (6/2) = 3

10 < 15

0	1	2	3	4	5	6
3	5	8	10	15	20	25

S=4 M=5 E=6

S = M + 1 = 3+1 = 4

M = (4+6)/2 = 5

M = 4 + ((6-4)/2)
= 4 + (2/2)
= 4 + 1 = 5

20 > 15

0	1	2	3	4	5	6
3	5	8	10	15	20	25

S=4
E=4
M=4

E = M - 1 = 5 - 1 = 4

M = (4+4)/2 = 4

M = 4 + ((4-4)/2)
= 4

Return 4;

15 = 15

BINARY SEARCH 1

Pseudocode

```
int binarySearch (int arr[], int T) {

    int S = 0;
    int e = arr.length - 1;

    while (S <= e) {                    // Loop will stop if S > e

        mid = (S + e) / 2;              // mid = S + ((e - s)/2)

        if (arr[mid] == T) { return mid; }
        else if (arr[mid] > T) { e = mid - 1; }
        else { S = mid + 1; }
    }

    return -1;

}
```

O(log N) — while loop; O(1) — body of the loop

Time Complexity — O(log n)
Space Complexity — O(1)

Note: As we know, whenever we have a loop in an answer, we can also solve a problem with the help of recursive calls. But we will not use recursion to solve any binary search problems because it increases space complexity to O(log n) from O(1) in simple loop approach.

BINARY SEARCH 1

Binary Search on an array with duplicates

Suppose we have following array

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	3	3	3	3	4	5	8	10	10

S=0 M=6 E=12

Finding first occurrence

S	E	M	Ans
0	12	6	6

Target 3

✡ Here we found the target at 6th index but we want to return 3 as index the answer which has first occurrence of 3 in the array.

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	3	3	3	3	4	5	8	10	10

S=0 M=2 E=5

S	E	M	Ans
0	5	2	6

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	3	3	3	3	4	5	8	10	10

S=3 M=4 E=5

S	E	M	Ans
3	5	4	4

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	3	3	3	3	4	5	8	10	10

S=3
E=3
M=3

S	E	M	Ans
3	3	3	3

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	3	3	3	3	4	5	8	10	10

E=2 S=3

S > E, Terminate

S	E	M	Ans
3	2		

S > E, Terminate

BINARY SEARCH 1

Pseudocode - Binary search on an array with duplicates. Finding first occurrence

```
int binarySearch (int arr [], int T) {

    int s = 0;
    int e = arr.length - 1;
    int ans = -1;

    while (s <= e) {

        mid = (s + e) / 2;

        if (arr [mid] == T) { ans = mid; e = mid - 1; }
        else if (arr [mid] > T) { e = mid - 1; }
        else { s = mid + 1; }

    }

    return ans;

}
```

BINARY SEARCH 1

Pseudocode - Binary search on an array with duplicates — Finding last Occurrence

```
int binarySearch (int arr[], int T) {

    int S = 0;
    int e = arr.length - 1;
    int ans = -1;

    while (S <= e) {

        mid = (S + e) / 2;

        if (arr[mid] == T) { ans = mid; S = mid + 1; }
        else if (arr[mid] > T) { e = mid - 1; }
        else { S = mid + 1 }

}
```

Conditions for when to apply Binary Search in a Data Structure:

1. The data structure must be sorted
2. Access to any element of the data structure takes constant time

IMPORTANT NOTE: It is ~~not~~ necessary that we will ~~be~~ always be provided with a number to find in sorted array. Sometimes, there could be certain other conditions that are available in question. We need to use these conditions to find the answer. NEXT QUESTION is one such problem which doesn't have a specified no. to find but there are some conditions.

BINARY SEARCH 1

Qw 1 There is a sorted array which is K rotated. ① We need to find the pivot element. ② We need to find out the index of any element in Array of **unique** elements

Check Question 7 too

K Rotated means

1	2	3	4	5	6	7

4	5	6	7	1	2	3

K^{th} rotated array

Either of them could be pivot element

Ans **Approaches**

① Linear Search $O(n)$

② Binary Search $O(\log n)$

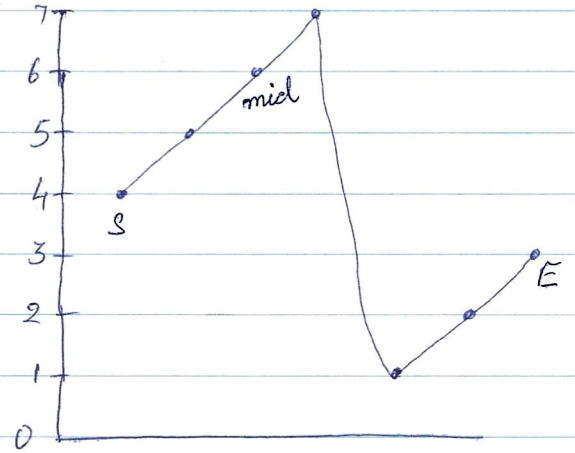
Binary Search

① Pivot element could be found through binary search. $O(\log N)$

② Once we know the pivot element we could find the index of certain element again by binary search. $O(2 \log n)$ = $O(\log n)$

This element will be provided in the question.

Case 1

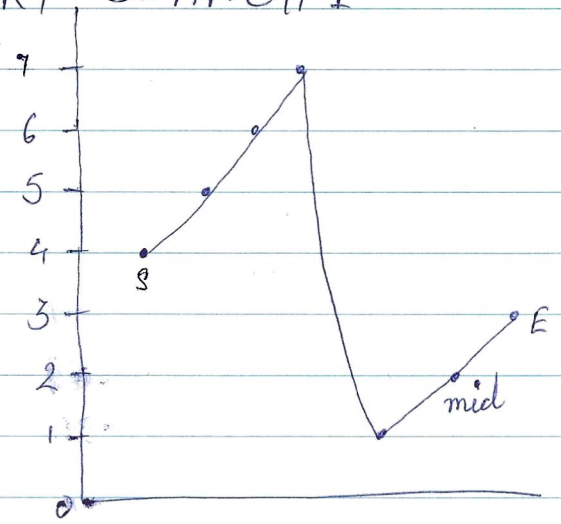


~~if (mid > S) S = mid + 1;~~

```
if(arr[mid] > arr[S])
    S = mid + 1;
```

BINARY SEARCH 1

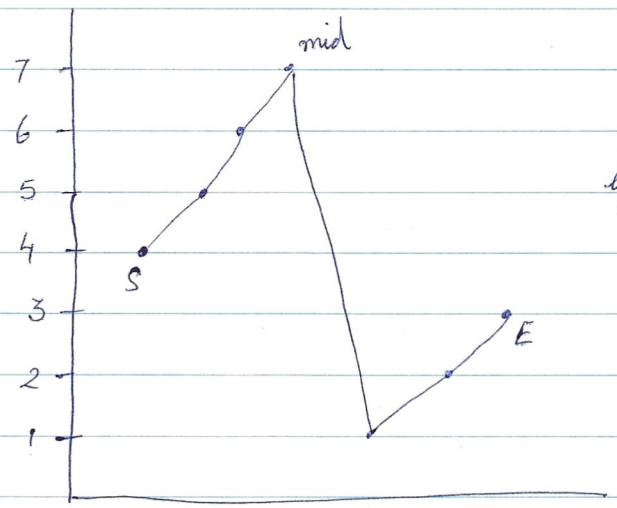
Case 2



~~if (mid < S) E = mid-1~~

```
if (arr[mid] < arr[S])
    E = mid - 1
```

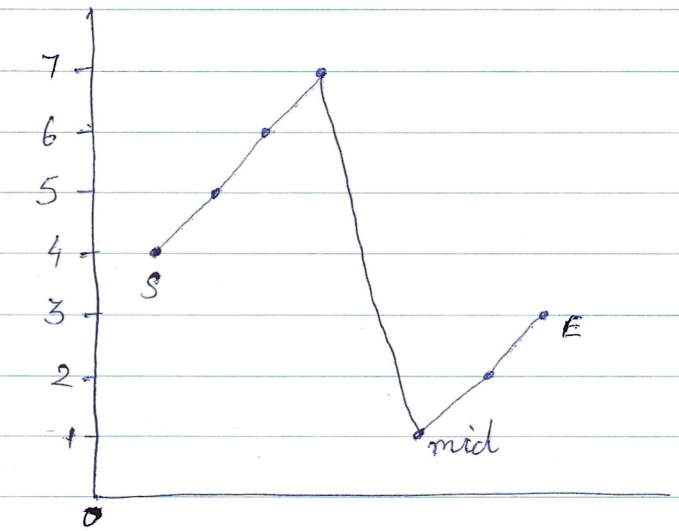
Case 3



~~if (mid > mid+1) return mid+1~~

```
if (arr[mid] > arr[mid+1])
    return arr[mid+1]
```

Case 4



~~if (mid < mid-1) return mid~~

```
if (arr[mid] < arr[mid-1])
    return arr[mid]
```

We can use above 4 cases to define the binary search conditions to find out the pivot point.

IMPORTANT: If the target is not present in the array and we want to find the potential index where it could be placed then in a general binary search algorithm start variable in the end bears the potential index where it could be placed.

BINARY SEARCH 1

Question 2 Find the expected index of a number in sorted array if it doesn't exist else return the index, if it is present. TC should be O(Log n)

Approaches

1. Linear Search TC- O(N), SC - O(1)
2. Binary Search TC - O(Log n), SC - O(1)

Corner Cases 1

arr = 1, 2, 3, 4, 6, 7, 8 (indices 0 1 2 3 4 5 6)

target = 0

0	1	2	3	4	5	6
1,	2,	3,	4,	6,	7,	8
↓			↓			↓
S=0			M=3			E=6
			4>0			

S=0 M=1 E=2

2>0

S=0
E=0
M=0
1>0

E=-1, S=0 (Terminate)

Answer = S

Corner Case 2

arr = 1, 2, 3, 4, 6, 7, 8 (indices 0 1 2 3 4 5 6)

target = 9

0	1	2	3	4	5	6
1,	2,	3,	4,	6,	7,	8
S=0			M=3			E=6
			9>4			

S=4 M=5 E=6

9>7

S=6
E=6
M=6
9>8

S = 6+1 = 7
E = 6 (Terminate)

Answer = S

IMPORTANT: Try to plot a graph for the problem to understand the ask of the question and solve it.

BINARY SEARCH 1

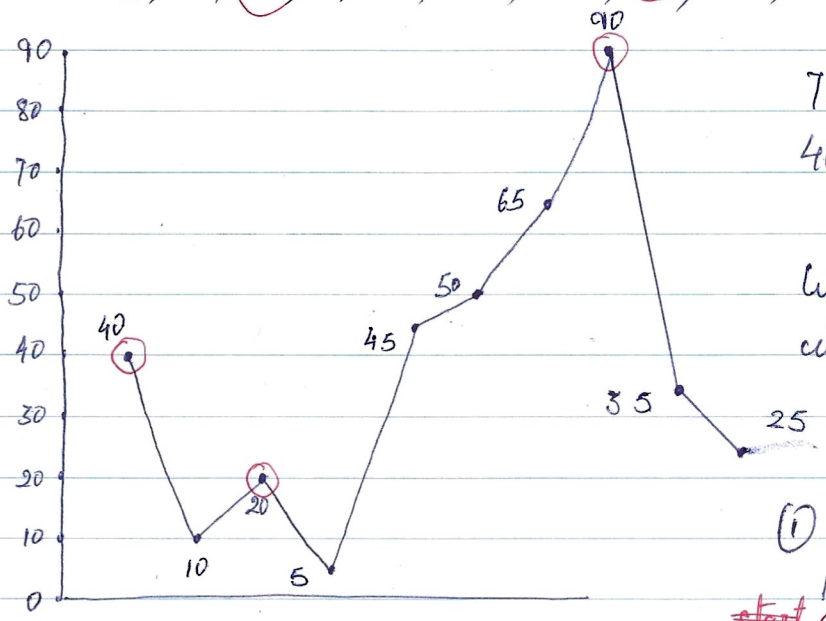
Question 3 Given an array, we need to find the peek element in the array. A peek element is an array element that is greater than it's neighbours. There could be deplicates in the array. There could be multiple peek elements in an array. In that case, we can return only one of them.

Approaches

1. Linear Search TC - O(n), SC = O(1)
2. Binary Search TC = O(log n), SC - O(1)

Binary Search — Intuition Building.

Arr = [40, 10, 20, 5, 45, 50, 65, 90, 35, 25]



There are 3 peek elements.
40, 20, 90

We need to write algorithm in a way that flow should move toward peek element

1. If we are at (mid) 10 then we should move toward 40 because 40 > 10
   End = mid - 1 (start)
2. If we are at (mid) 50 then we should move toward 65 because 65 > 50
   start = mid + 1

Corner Case 1

3. If we reach at 0th index 40 then we can return answer as 40

Corner Case 2

4. In the same way if have 46 as the last element after 25 then that could be returned as a peek element.

BINARY SEARCH 1

Question 4 Given two sorted arrays arr1 and arr 2 ~~of size~~. Return the median of two sorted arrays. #

Approaches

(a)

(1) Two pointer method to sort the array while merging them.
   (b) Thereafter find the median in even and odd case differently.
   (c) TC - O(M+N) , SC - O(M+N) [A new array will be created]

(2) Two pointer method
   (a) We will first find the total no. of elements that merged array could have but we will not create one.
   (b) Check even or odd length.
   (c) We will maintain the variables for both even length or odd length to record the indices value.
   (d) Iterate over both the arrays as we did in approach 1
   (e) If index counter matches the index needed to calculate median, we will store the arr[i] in ~~on a~~ variables
   (f) In the end, we will calculate the median and return it. TC - O(M+N) , SC - O(1)

(3) Binary Search.

Median even length - $\frac{(n/2-1)^{th}\text{ ele} + (n/2)^{th}\text{ ele}}{2}$

Median odd length - $(\text{Math.floor}(n/2)+1)^{th}$ ele

arr 1 = [7, 12, | 14, 15] -> 4
arr 2 = [1, 2, 3, | 4, 9, 11] -> 6
6+4 = 10 → EVEN

S = 0 E = 4

we can take minimum 0 and maximum 4 elements from arr 1

Even medianPos = $\frac{10}{2} = 5$

mid 1 = $\frac{S+E}{2} = \frac{0+4}{2} = 2$

mid 2 = 5 - 2 = 3

L1 = 12 R1 = 14

L2 = 3 R2 = 4

L1 (12) < R2 (4) ✗

E = mid 1 - 1

= 2 - 1 = 1

condition

arr 1 = [| 7, 12, 14, 15]

arr 2 = [1, 2, 3, 4, 9, | 11]

S = 0 E = 1

mid 1 = $\frac{0+1}{2} = \frac{1}{2} = 0$

mid 2 = 5 - 0 = 5

L1 = int-min R1 = 7

L2 = 9 R2 = 11

min L1 < R2 (11) ✓

L2 (9) < R1 (7) ✗

S = mid 1 + 1

= 0 + 1 = 1

condition

arr 1 = [7, | 12, 14, 15]

arr 2 = [1, 2, 3, 4, | 9, 11]

S = 1 E = 1

mid 1 = $\frac{1+1}{2} = 1$

mid 2 = 5 - 1 = 4

L1 = 7 R1 = 12

L2 = 4 R2 = 9

L1 (7) < R2 (9) ✓

L2 (4) < R1 (12) ✓

Condition Satisfied

Even Median = $\frac{\text{Math.max}(L1, L2) + \text{Math.min}(R1, R2)}{2}$

NOTE: Out of 2 sorted arrays provided use the smaller array to run binary search on.

IMPORTANT: Always try to decrease the search space.

BINARY SEARCH 1

arr 1 = [7, 12, | 14, 15, 16] → 5
arr 2 = [1, 2, 3, 4, | 9, 11] → 6

5+6 = 11 ODD

Odd medianPos = $\frac{11+1}{2} = \frac{12}{2} = \boxed{6}$

S = 0 E = 5

$mid1 = \frac{S+E}{2} = \frac{0+5}{2} = \frac{5}{2} = 2.5 = \boxed{2}$

mid 2 = 6 - 2 = 4

L1 = 12 R1 = 14
L2 = 4 R2 = 9

L1 < R2 ✗
12 9
E = mid 1 - 1 = 2 - 1 = 1

condition

arr 1 = [| 7, 12, 14, 15, 16]
arr 2 = [1, 2, 3, 4, 9, 11 |]

S = 0 E = 1

$mid1 = \frac{0+1}{2} = .5 = \boxed{0}$

mid 2 = 6 - 0 = 6

L1 = int_min R1 = 7
L2 = 11 R2 = int_max

L1 < R2 ✓
min max
L2 < R1 ✗
11 7
S = mid 1 + 1 = 0 + 1 = 1

condition

arr 1 = [7, | 12, 14, 15, 16]
arr 2 = [1, 2, 3, 4, 9, | 11]

S = 1 E = 1

$mid1 = \frac{1+1}{2} = 1$

mid 2 = 6 - 1 = 5

L1 = 7 R1 = 12
L2 = 9 R2 = 11

L1 < R2 ✓
7 11
L2 < R1 ✓
9 12

Condition satisfied

Odd Median = Max (L1, L2)

BINARY SEARCH 1

Question 2.15 Find the number or index of the number which is just greater than the given number.

arr = 1, 3, 5, 5, 5, 6, 6

num = 5

```
 0  1  2  3  4  5  6
 1, 3, 5, 5, 5, 6, 6
S=0      Mid=3      E=6
```

STEP 1

$Mid = \frac{S+E}{2} = \frac{0+6}{2} = 3$

arr [3] $\not<=$ 5

arr [mid] $\not>$ num

3 $\not>$ 5

S = mid + 1

= 3 + 1 = 4

```
 0  1  2  3  4  5  6
 1, 3, 5, 5, 5, 6, 6
          ↓  ↓  ↓
          S=4 Mid=5 E=6
```

STEP 2

$Mid = \frac{S+E}{2} = \frac{4+6}{2} = 5$

arr [mid] > num

arr [5] > 5

6 > 5

E = mid - 1 = 5 - 1 = 4

```
 0  1  2  3  4  5  6
 1, 3, 5, 5, 5, 6, 6
          S=4
          E=4
          Mid=4
```

STEP 3

$Mid = \frac{S+E}{2} = \frac{4+4}{2} = 4$

arr [mid] $\not<=$ num

arr [4] $\not>$ 5

5 $\not>$ 5

S = mid + 1 = 4 + 1 = 5

```
 0  1  2  3  4  5  6
 1, 3, 5, 5, 5, 6, 6
          ↓  ↓
          E=4 S=5
```

STEP 4

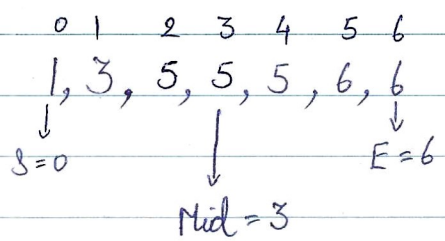
S > E Terminate

return S;

BINARY SEARCH 1

arr = 1, 3, 5, 5, 5, 6, 6

num = 4



STEP 1

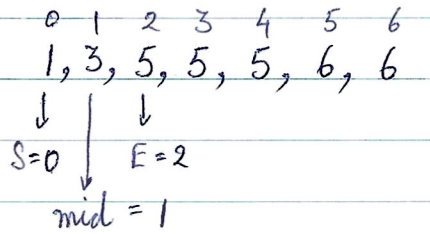
$mid = \frac{S+E}{2} = \frac{0+6}{2} = 3$

arr [mid] > num

arr [3] > num

5 > 4

E = mid - 1 = 3 - 1 = 2



STEP 2

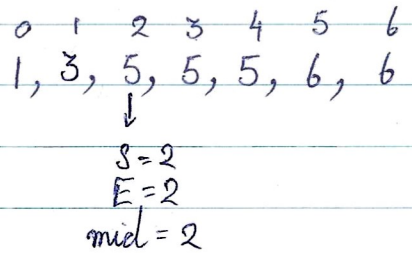
$mid = \frac{0+2}{2} = 1$

arr [mid] $\not>$ num

arr [1] $\not>$ 4

3 $\not>$ 4

S = mid + 1 = 1 + 1 = 2



STEP 3

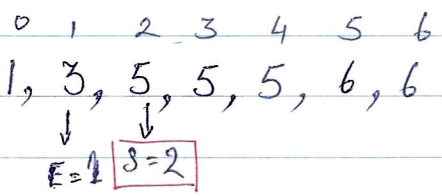
$mid = \frac{2+2}{2} = 2$

arr [mid] > num

arr [2] > 4

5 > 4

E = mid - 1 = 2 - 1 = 1



STEP 4

S > E

2 > 1 Terminate

Return S;

BINARY SEARCH 1

Question 6 Given a matrix of integers A of size N×M in which each row is sorted. Find and return the overall median of matrix A.

N×M is always odd

Note: No extra memory is allowed

Note: Rows are numbered from top to bottom and columns are numbered from left to right.

Approaches

With extra Space

(1) We can make an extra array to put all the elements of matrix into it. TC - O(m×n)

(2) After that we will sort the array. TC - $\frac{}{}$m.n O(m.n)

(3) We will find out the median from the sorted array.

(4) TC - m.nO(m.n) SC = O(m.n)

Without extra Space Binary Search

(1) Firstly, we will find the median position (index in array)

(2) We will fetch the smallest and the highest no. in the matrix.

(3) Now, we will run a binary search using smallest and highest no.

(4) In the binary search, we will make comparison between the mid and median position calculated in step 1.

no. of elements smaller than

$10^9 = 2^{32}$ — Max range of integer

BINARY SEARCH 1

(5) To calculate the no. of elements smaller than mid, we again need to write a binary search inside the previous binary search loop.

(6) TC — $\log_2(2^{32}) \times N \times \log_2 M$ SC = O(1)

$= 32 \times N \times \log_2 M$

Example: n=3

arr =

	0	1	2	
0	2	3	5	
1	2	3	4	m=3
2	1	7	9	

min smallest no = 1

max highest no = 10^9

median Pos = m × n / 2

OR

min smallest no = 1 (smallest no. in matrix)

max highest no = 9 (highest no. in matrix)

median Pos = m × n / 2 = 3×3/2 = 9/2 = 4

STEP 1: min = 1, max = 9, mid = 1+9/2 = 10/2 = (5)

» No. of elements smaller than equal to 5 are 1, 2, 2, 3, 3, 4, 5 which are 7

Since 7 > 4 (medianPos), max will be updated to mid - 1

max = mid - 1 = (5) - 1 = 4

STEP 2: min=1, max = 4, mid = 1+4/2 = 5/2 = (2)

» No. of elements smaller than equal to 2 are 1, 2, 2 which are 3

Since 3 < median Pos [4], min will be updated to mid+1

min = mid + 1 = (2) + 1 = 3

BINARY SEARCH 1

STEP 3: min = 3, max = 4, mid = 3+4/2 = 7/2 = 3

» No. of elements smaller than equal to 3, 1, 2, 2, 3, 3 which are 5

medianPos

Since 5 > 4, max will be updated to mid - 1

max = mid - 1 = 3 - 1 = 2

STEP 4: min = 3, max = 2

min > max TERMINATE

ANSWER = min = 3

medianPos 4

Difference of way of calculation of ~~mid~~ for question 5 and 6 in case of odd no. of elements in search space in question

» In que 5, we find the no. of elements <= medianPos

Que 7 There is a sorted array which is K rotated. We need to find out the index of any element in Array of duplicate elements. Check Question 1 too. • If element exists return true otherwise false.

K rotated means

0	0	1	2	2	5	6

2	5	6	0	0	1	2

target = 0

Ans Approaches

① Linear Search - TC - O(n)

② Binary Search - TC - O(log n) for the best and average case. O(n/2) for the worst case. Here, N = size of the given array.

Reason: In the best and average scenarios, the binary search algorithm is primarily utilized and hence the time complexity is O(log N).

BINARY SEARCH 1

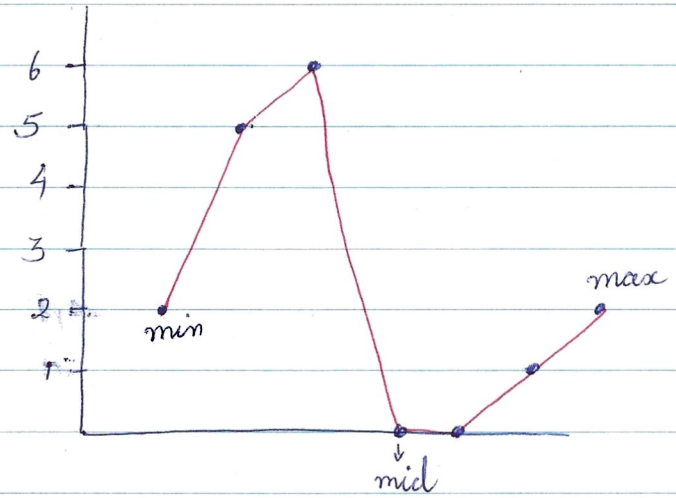
However, in the worst-case scenario, where all array elements are the same but not the target (eg. given array = [3, 3, 3, 3, 1, 3, 3], target = 1), we continue to reduce the search space by adjusting the low and high pointers until they intersect. This ~~worst~~ worst situation incurs a time complexity of $O(N/2)$.

Best and average case

min = 0, mid = 3, max = 6

[2, 5, 6, 0, 0, 1, 2]

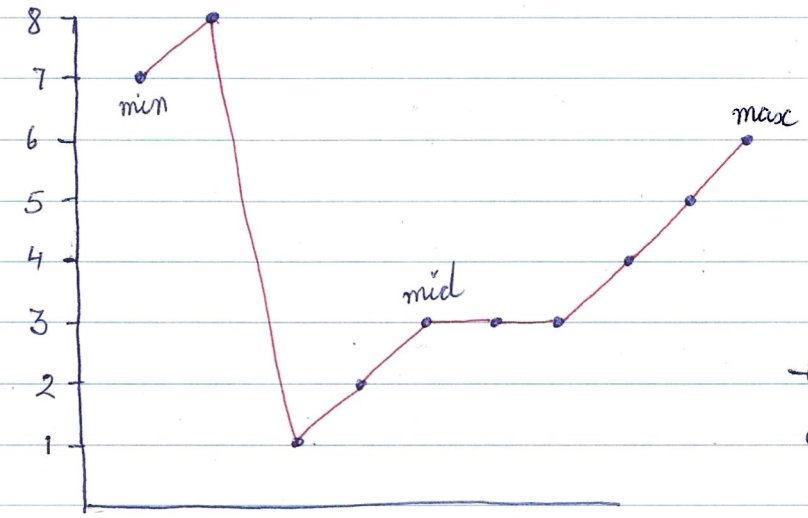
target = 0



Best Case

Answer = True

arr[mid] == target



Average Case

min = 0, mid = 4, max = 9

[7, 8, 1, 2, 3, 3, 3, 4, 5, 6]

target = 3

~~arr[mid] ← arr[max]~~

arr[mid] == target

Answer = True.

BINARY SEARCH 1

Worst Case

min mid max
0 3 6
[3, 1, 2, 3, 3, 3, 3]
Target = 1

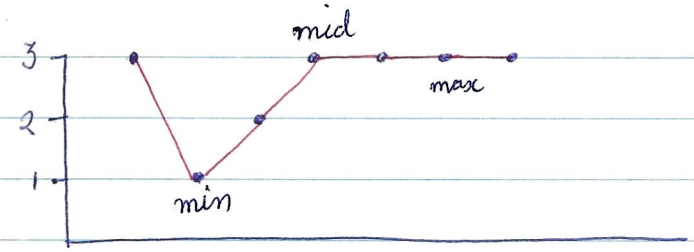
arr[min] == arr[mid] == arr[max]

Extra Step

min++;

max--;

min mid max
1 3 5
[3, 1, 2, 3, 3, 3, 3]



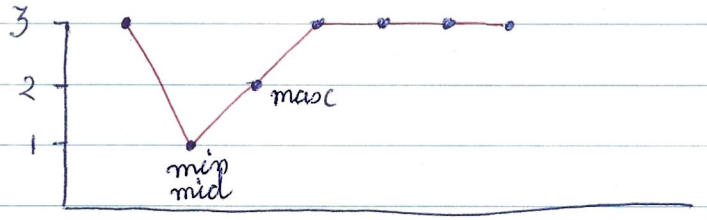
arr[min] < arr[mid]

~~target >= arr[min] &&~~

arr[min] <= target <= arr[~~max~~ mid]

max = mid - 1

mid
min max
1 2
[3, 1, 2, 3, 3, 3, 3]

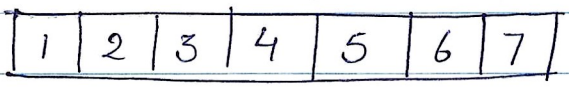


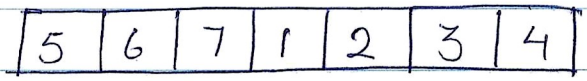
arr[mid] == target

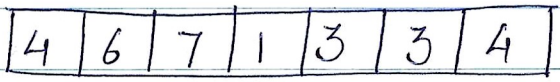
return true;

BINARY SEARCH 1

Question 8 Find minimum in Rotated sorted array with duplicate elements.



Rotated sorted array = 

Rotated sorted array with duplicate elements = 

Answer Approaches

① Linear Search – O(n) – TC
② Binary Search – TC – O(log n) in Average case
O(1) in best case
O(n/2) in ~~worst~~ worst case.

Average Case

```
          0  1  2  3  4  5  6
arr = [ 3, 4, 4, 6, 7, 1, 3 ]
```

→ min = 0, max = 6, mid = 3, answer = int.max

arr[mid] >= arr[min] Condition 3

~~e~~ answer = arr[min];
min = mid + 1;

3 < int.max

→ min = 4, max = 6, mid = 5, answer = 3

arr[mid] <= arr[max] Condition 4 else

answer = arr[mid]
max = mid - 1;

BINARY SEARCH 7

$1 < 3$

→ min = 4, max = 4, mid = 4, answer = 1

arr [min] == arr [mid] == arr [max] Condition 2

```
min++;
max--;
answer = min(answer, arr[mid]);
```

→ min = 5, max = 3,

min > max Terminate.

return answer;

Best Case — Array is not rotated at all

arr = [1, 3, 5];
(0, 1, 2)

→ min = 0, max = 2, mid = 1, answer = int.max

arr [min] <= arr [max] Condition 1

```
answer = arr[min];
break;
```

return answer;

BINARY SEARCH 1

Worst Case

	0	1	2	3	4	5	6
arr =	3,	3,	3,	3,	1,	3,	3

→ min = 0, max = 6, mid = 3, answer = int.max

arr[min] == arr[mid] == arr[max]
min++;
max--;

→ min = 1, max = 5, mid = 3, answer = int.max

arr[min] == arr[mid] == arr[max]
min++;
max--;

→ min = 2, max = 4, mid = 3, answer = int.max

~~arr[min] <= arr[mid]~~
~~answer = arr[min];~~

arr[mid] >= arr[min]
answer = arr[min];
min = mid + 1;

→ min = 4, max = 4, mid = 4, answer = 3

arr[min] == arr[mid] == arr[max]
answer = arr[mid]; // 1 < 3
min++;
max--;

→ min = 5, max = 3 min > max, terminate.

return answer;

BINARY SEARCH 1

Question 1 You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.

Return the single element that appears only once.

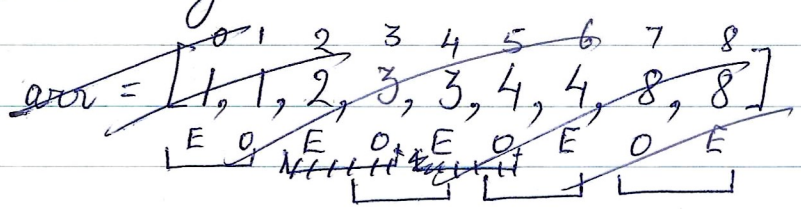
Your solution must run in O(log n) time and O(1) space.

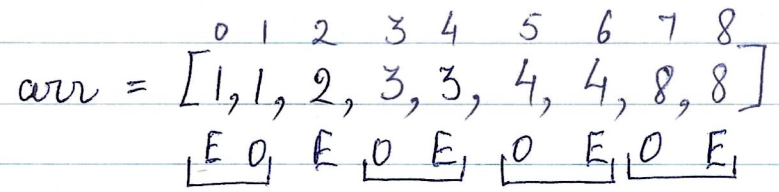
Eg: arr = [1,1,2,3,3,4,4,8,8]

Answer

Approaches

1. Linear Search - TC = O(n), SC = O(1)
2. Linear Search with hash map - TC = O(n), SC = O(n)
3. Binary Search:





→ min = 0, max = 8, mid = 4

arr[mid] == arr[mid-1] && mid-1 % 2 != 0

max = mid - 1

BINARY SEARCH 1

→ min = 0, max = 3, mid = 1

arr[mid] == arr[mid-1] && mid-1 % 2 == 0

min = mid + 1

→ min = 2, masc = 3, mid = 2

arr[mid] != arr[mid-1] && arr[mid] != arr[mid+1]

return arr[mid];

	0	1	2	3	4	5	6	7	8
arr =	1,	1,	2,	2,	3,	3,	4,	5,	5]

~~O E O~~

E O E O E O E O E

→ min = 0, max = 8, mid = 4

arr[mid] == arr[mid+1] && mid+1 % 2 != 0

min = mid + 1

→ min = 5, masc = 8, mid = 6

arr[mid] != arr[mid-1] && arr[mid] != arr[mid+1]

return arr[mid];